

パワープラント制御ソフトの自動生成

Autocoding of Power Plant Control Software

箕輪 信太郎*
Shintaro MINOWA
石黒 哲矢*
Tetsuya ISHIGURO

廣田 俊明*
Toshiaki HIROTA
大塚 訓*
Satoru OHTSUKA

横山 浩紀*
Hiroki YOKOYAMA

要 旨

制御系設計 CAD 上で ECU 実装時相当の固定小数点演算のシミュレーションをおこない、設計した制御仕様を自動ソフト生成するシステムの構築をおこなった。同システムを用いることにより、ソフト生成時に必要となる変数の精度(LSB)、データ型等の設定を容易におこなえるものとした。生成されるソフトを Honda 書式にすると共に、実車テストにおける変数セッティングやモニタに必要となる変数情報の自動生成をおこなうものとした。その結果、自動生成したソフトに一切変更を加えることなく ECU に実装をおこなうことができた。またカバレッジチェックによる演算の検証により、ECU実装での動作不良を未然に防ぐことができた。これらによる戻り工数の削減、及びソフトの自動生成により、ECU実装期間を約3分の2に短縮し、1回の実装で不具合を発生することなくシミュレーションと同等の動作を実現できた。

ABSTRACT

An autocoding system has been developed for use in CAD of control systems. This system conducts simulations of fixed point calculations equivalent to those used in ECU implementation, and autocodes the designed control specifications. The use of this system enables simple formulation of the data accuracy (LSB) and data type required for software coding. In addition to coding software in a Honda format, the system autocodes the label information required in setting parameters and monitoring for vehicle tests. As a result, the autocoded software can be employed in ECU implementation without alteration. Verification of the accuracy of calculations using a coverage check enables ECU malfunctions to be prevented before implementation. This has reduced the time spent in correcting the software, and autocoding has shortened the period to ECU implementation by one-third. A level of performance equivalent to the simulation has been achieved with no malfunctions in a single implementation.

1. はじめに

パワープラントは、環境適合、燃費向上、自己故障検知等、複雑・大規模かつ高精度な制御を必要としている。しかし一方で開発期間の短縮、開発機種の増加等、開発の効率化が重要な課題となっている。そこでパワートレインECUソフトウェアの品質を確保し、開発効率を向上させるために、ソフトウェアのライブラリ化の適用が開始された。これによりハードウェアに依存する部分とアプリケーション部分は分離され、アプリケーションは既存のソフトを組合わせた開発が可能となった。

アプリケーションはシミュレーションツール(MATLAB, Simulink)により設計し、確立した制御ロジックを制御仕様書にしてハンドコーディングにて Honda 書式としたが、
(1)ハンドコーディングによる設計意図の不整合
(2)シミュレーションと実装時の実演算との不一致

等の問題点があった。これらの不整合を生じない手法の実現方法として、シミュレーションツール上にて実装時の演算を Honda 書式でおこない、その結果を自動ソフト生成するシステムの開発をおこなった。

2. システム開発のねらい

ソフトの自動生成の目的は以下となる。

- (1)ロジック設計者とプログラマ間のロジック解釈不一致の排除
- (2)ソフト記述ミスの低減
- (3)作業の自動化による開発工数の削減

2.1. 開発工程の短縮

ECUでは処理速度、メモリ容量等の観点から、一般に用いられている浮動小数点演算ではなく、固定小数点による

演算をおこなっている。Fig. 1に浮動小数点と固定小数点の波形の比較を示す。固定小数点演算では浮動小数点演算と異なり、変数の大きさとLSBを適正に設定しなければ必要な演算精度を得ることができない。従来の開発工程では浮動小数点での動作確認を終えたモデル図をソフト生成(ハンドコーディング)対象とするため、固定小数点演算は実装後に初めておこなわれ、発見できた動作不整合を改修するためにソフト修正が発生していた。本システムでは制御ロジック設計ツールは、制御系設計CAD(Simulink)⁽¹⁾と、汎用の整数型ソフト自動生成ツール(TargetLink)⁽²⁾を用いた。これはモデル図上で固定小数点演算をおこない、そのロジックのソフトを自動生成するツールである。本システムを適用した場合には、一連の戻り工数がPC上で完結できるため、ECUを開発する上で工数削減が望める。Fig. 2に従来のECU開発工程と本システムを適用した工程との比較を示す。

2.2. 修正工程無しの開発環境

汎用の整数型ソフト自動生成ツールではANSI-C, Target (ECUチップ)に合わせた最適ソフト生成機能を有するが、Honda書式とは異なるため、そのままではソフトの実装に手修正が必要となる。Hondaでは常用されるセンサ出力等の変数はラベル名, LSB, データ型等を共有の情報として独自ライブラリに登録、管理しているため、これとリンクしたシステム、かつロジック構築からソフト生成まで修正工数が発生しない開発環境として以下の機能を作成した。

- (1) ライブラリから変数情報を取得し、モデルに反映する機能
- (2) 固定小数点演算の変数形式を Honda 書式にする機能

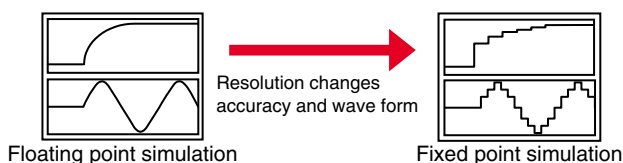


Fig. 1 Comparison of floating point and fixed point wave form

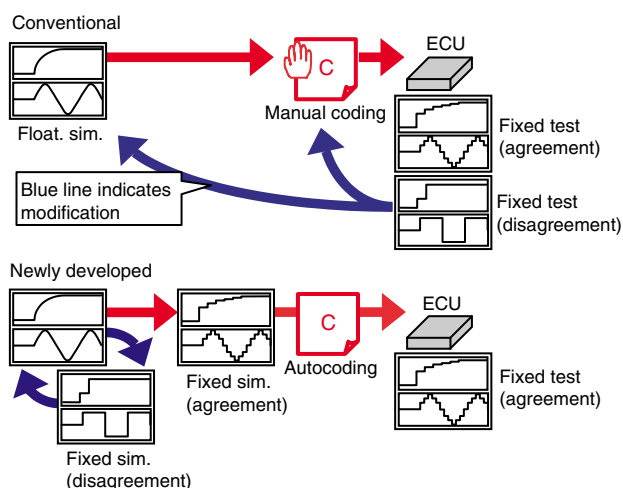


Fig. 2 Comparison of ECU design processes

- (3) カバレッジ検証による動作確認機能
- (4) Honda 書式に適合した自動ソフト生成機能
- (5) 従来仕様書に代わるドキュメント生成機能

Fig. 3 に開発システム概要を示す。これにより固定小数点演算時及び、自動生成ソフトの実装時等に手修正が不要となる。

3. モデル図の Honda 書式化

本システムの機能について以下に述べる。

3.1. 変数情報設定機能

ライブラリは変数情報だけでなく、その変数を使用する制御ロジックの仕様書とそれを基にしたソフトも登録してあるため、新たに制御ロジックを作成する際に、用途に応じて制御ロジック、ソフトとも再利用可能である。ライブラリではロジックに使用する変数情報をCSVファイルとして一元管理、共有化をおこない、異なるロジックであっても同じ情報を扱う変数は同一ラベル名を用いている。これによりロジック毎にラベルの変数情報の設定をおこなうことなく、LSB, データ型等を設定可能となる。また元となるライブラリの変数情報が変更(更新)された場合にも、その変数を用いる全てのロジックに変更の反映がおこなわれることになる。これと本システムによる変数情報の設定を合わせるため、以下の機能を作成した。

- (1) モデル図作成時にライブラリの変数情報を参照する機能
- (2) ライブラリの変数情報更新時にモデルの変数情報を更新するための機能
- (3) 新規に変数情報を作成し、ライブラリに登録する機能

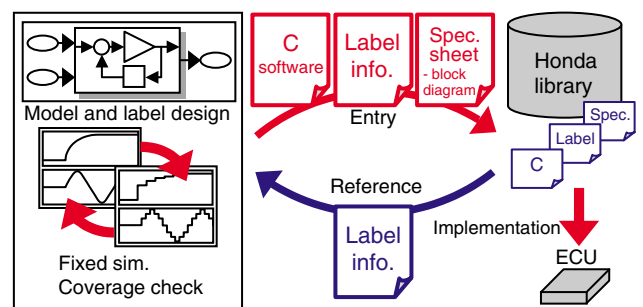


Fig. 3 Newly developed process

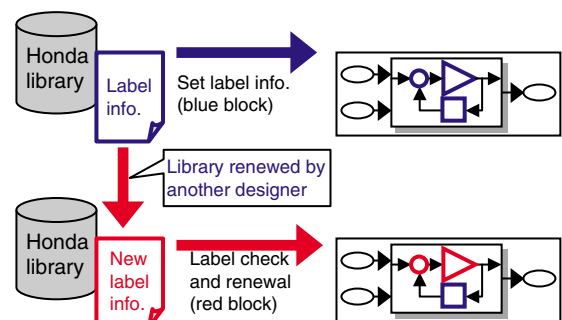


Fig. 4 Label information check and renewal tool

これにより新規モデル作成時においても既存のラベルに対しては、LSB 等の変数情報の自動設定が可能となり、その変数情報をライブラリの最新情報へ自動更新することができる。また変数情報自動更新機能は、モデル中で新規作成したラベルが既存の変数情報と重複する危険性の回避にも利用できる。同機能を Fig. 4 に示す。

3.2. 変数属性自動切替

汎用の整数型ソフト自動生成ツールによる固定小数点演算では、ロジックで用いる変数が必ずロジック内で定義されている必要がある。しかし実際のロジックでは、センサの信号値等が外部変数で与えられる場合があり、その際は演算をおこなう度に変数属性の宣言を変更する必要があった。これをシミュレーション時は定義(Global)変数とし、ソフト自動生成時は参照(Extern)変数とするような、自動生成の設定に合わせて変数属性の宣言を自動で切り替える変数の型を作成し、ソフト生成時の変数属性のままで固定小数点演算をおこなえるものとした。Fig. 5 に変数属性自動切り替え機能の概要を示す。

3.3. Map 検索ルーチンの置換

通常 Map, Table を使用する際は汎用の整数型ソフト自動生成ツールに用意されているブロックを使用する。しかし Honda では Map 検索ルーチンに独自のものを規定して用いているため、汎用の整数型ソフト自動生成ツールのブロックをそのまま使用することができない。そこで検索ルーチンを置き換え、ソフト自動生成時に Honda 書式での出力をおこなう機能を作成した。検索ルーチンは独自のものとなるが、モデル作成時に使用するブロックは汎用の整数型ソフト自動生成ツールと同じものを用いているため、制御ロジック設計者は意識することなく置き換えをおこなえるものとしている。ソフト生成時だけでなく、制御ロジック検討演算時より標準の検索ルーチンからの置き換えを図り、実装時との動作の同一性を持たせている。

3.4. データ型の規定

一般的な変数がライブラリとして共有化されているのと同様に、Honda で使用するデータ型も規定している。汎用の整数型ソフト自動生成ツールでは汎用性を持たせるため、非常に多くのデータ型を標準で持っており、型の選択時に人的ミスを生じる可能性がある。そこで本システムに

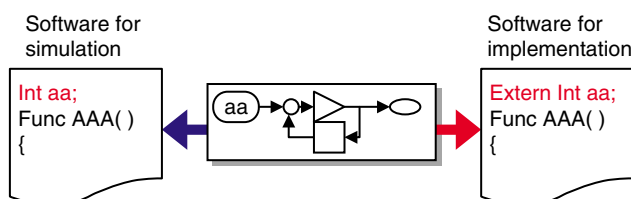


Fig. 5 Auto switch of C software generation

より Honda 独自のデータ型のみ選択する形式とした。また使用しているデータ型がわかるように、変数名に型の情報を付加する。これは従来は手動でおこなう部分であったが、データ型選択時に自動付加することでミス低減と効率化を図っている。

3.5. 初期化宣言

制御ロジックを構築する際に、エンジン始動、停止等の特定の動作モードにおいて、変数の初期化が必要となる場合がある。しかし初期化の方法は各社独自の手法をとっており標準化されていないため、汎用の整数型ソフト自動生成ツールにもその機能は用意されていない。そこで既存の変数定義ブロックを用いて初期化宣言ブロックを作成し、ソース生成時に初期化ロジックをライブラリのルールに従ったCの関数として自動で付加する機能を作成した。

3.6. 既存ソフト取り込み

制御ロジックを構築する際に、既存のソフトの出力値を入力とするロジックの開発をおこなう場合がある。既存のソフトはモデル図が無く、ハンドコーディングによるソフトのみ存在する場合があり、これをロジック開発に用いるにはソフトの内容を理解し、モデル図に作成し直す必要があった。そこでソフトをそのままモデル図の一ブロックとして取り込む機能を作成した。Fig. 6 に同機能の概要を示す。これは対象となるソフトがライブラリで規定したコーディングルールに従っていることを利用して自動でブロックの入出力を生成し、モデル図のブロックとしてソフトを取り込む。またライブラリと連動することにより、取り込んだソフトの各種設定値を変更させて動作することを可能とした。

3.7. モデルライブラリ

カウンタ、スイッチ等、モデル図を構築する上で使用頻度の高いブロック構造が存在するが、これまでは各設計者

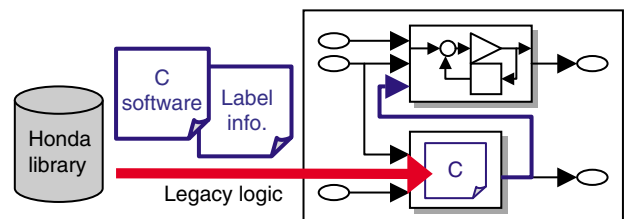


Fig. 6 Embedded function of legacy C software to new system

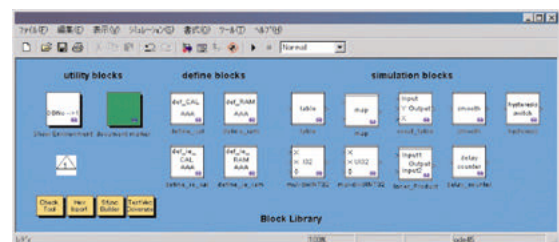


Fig. 7 New system model library

がその都度それらのモデル図を作成していた。そこでこれを予め作成し、サブシステム化してモデルライブラリとした。Fig. 7にモデルライブラリを示す。これによりモデル化工数を削減できるだけでなく、モデルライブラリ化された部分に対しては動作確認をおこなう必要が無く、モデル図記述も統一できる。また、本システムで作成した各機能も同ライブラリから容易に呼び出せる。

4. 動作確認

動作確認は任意入力データを用意することでおこなう。一般的には実テストによる測定データを入力データとする場合が多い。また自動ソフト生成では基本的に人間によるソフトの確認をおこなわず、LSB設定も人による継承をおこなったブロック以外は自動でおこなわれるため、設定をおこなう際に用いた入力データの重要性が必然的に高くなる。つまり実際に使用される際のデータの入出力値がLSB設定時の入出力値を上回る、或いはその値の組み合わせがテストされていない場合、実車テストや市場に製品が出てから初めてオーバーフロー等の検証ができていなかったことによる問題を生じる恐れがある。同様にソフト内に存在するif分岐、フラグによる動作切り替え等による全分岐の確認がおこなわれていなかった場合、その分岐に至るまでのロジックのオーバーフローの確認のみならず、ロジックの正当性やロジックそのものの必要性が検証されない可能性もある。そこでソフトの信頼性を向上するため、以下の機能を作成した。

- (1) 任意上下限値を持つ入力データ(テストベクタ)生成機能
- (2) 分岐のカバレッジ調査機能
- (3) 実行されていない分岐の強制切り替え機能

4.1. テストベクタ生成機能

テストベクタ生成機能はカバレッジ検証機能用の入力生成として作成した。本機能では振幅を任意に設定、変更できる2水準のデータを生成する。波形の形状は2水準データの組み合わせで表す総当たりと実験計画法(図中略:DOE)⁽³⁾の系統と、同データをパルス幅変調(PWM)波形として表す系統の二系統となる。Fig. 8に生成波形を示す。

4.1.1. 波形の種類

総当りは対象となる制御ロジックの入力端子数 n を因子数とした2水準の総当たりで、入力データを2水準と考えた場合の全組み合わせを網羅した入力である。しかしこれ

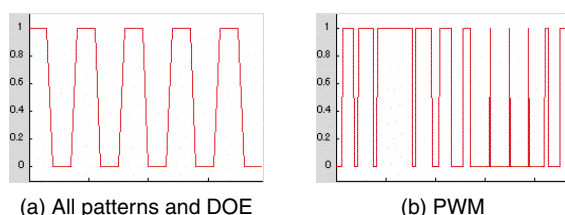


Fig. 8 Shape of test vector

では入力端子数が増えるほど生成される入力データも肥大化することを意味する。そこで入力データ量が少なくとも、総当たり相当の実験量を意味するデータの構築をおこなえる実験計画法を取り入れた。但し直交因子の選定は制御ロジックを構築する設計者に一任され、直交配列表すなわち入力データの基は制御系設計 CAD に用意されている関数を用いて作成する。

PWM 波形は入力端子毎に周波数及び周波数変化を既定値とし、これに基づくパルス幅変調による2水準データを生成する。生成波形は入力端子数分存在する PWM 波形の中で最長周期のデータ量に合わせ、他端子波形の出力周期を繰り返す形とする。

4.1.2. 繰り返し波形の振幅

現代制御で用いられる高次のフィルタや、多段に前回値保持をおこなうロジックでは、いかにデータを工夫してもフィルタ出力段でデータが丸められ、その先のロジック検証には至らない。或いはその次数が高次になるほど、根本的にデータ量が不足する等の問題も存在する。そこで上記入力波形を繰り返して生成する入力データを増幅する機能を作成した。Fig. 9に繰り返し回数5回の場合のテストベクタ波形を示す。繰り返しをおこなう際の振幅は以下の3種類を選択可能とした。

- (1) 任意に設定された各入力端毎の最大、最小値(図中a)
- (2) 繰り返し回数に応じて一定の振幅を維持したまま最大、最小値を変化させる(図中b)
- (3) 繰り返し回数に応じて、設定した最大、最小値を変化させていき、徐々に振幅を減少させる(図中c)

これによりオーバーフロー検出によるLSBのタフネス検証に併せ、実際のロジック動作上重要となる値近傍の動作確認も可能となる。

テストベクタ生成はモデルの入力端子数に合わせて入力データを生成する機能となっているが、モデル構成上ロジックに影響の無い値、フラグ情報で2水準の値等が存在する場合に、これらの入力端を因子の組み合わせから外すことで、更に入力データ量を抑制できる仕様とした。

4.2. カバレッジ検証機能

カバレッジ検証結果はテストベクタによるシミュレーション終了時に制御系設計 CAD により自動的に生成される。このときロジック内の分岐が全分岐を通っていない場合にカバレッジ結果が100%未満で表示される。そこでカバ

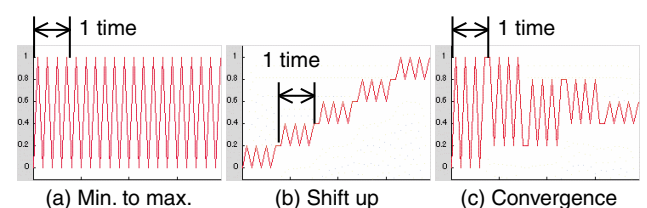


Fig. 9 Shape of test vector (5-time-repeat mode)

レッジ結果よりモデルの該当位置を表示し、分岐ブロックに関しては閾値を任意変更する機能を作成した。Fig. 10にカバレッジ検証フローを示す。これより1回目におこなったシミュレーションで分岐チェックが完了しなかった分岐ブロックを2回目のシミュレーションで分岐条件を変更し、最終的にカバレッジ 100%を得ている。

4.3. テストベクタ カバレッジ検証結果

Table 1にテストベクタ適用事例を示す。尚、適用対象モデルは分岐以外にもフィルタを持つ現代制御ロジックであり、実測値によるLSB 合わせを完了した状態である。またテストベクタの振幅は各入力データの物理的上下限値を用い、分岐切り替えを使用する前の結果である。実測値に対し、入力データ量が10分の1以下となるテストベクタではどれもオーバフローを検出しており、その有効性が認められる。テストベクタ同士を比較した際には、実験計画法がオーバフロー検出数では他に比して少ないが、入力データ量が極めて少なく、検出率は向上している。但しその後繰り返し数によりデータ量を増加させても検出数の向上には繋がらなかった。また直交因子の選択により結果には著しい差が生じ、因子選択そのものが効率化の鍵となり、必ずしも効率化の助けとは成り得ない場合も存在する。

テストベクタにより実測値で検証しきれなかったオーバフローの検出をおこなえ、モデルのタフネス向上を図ることができたが、カバレッジ向上は図れなかった。

5. ソフト生成

ソフト生成をおこなう際にはモデル図の各ブロックに付した変数情報の他に、従来手法では仕様書作成段階で記述する作者、作成日、改訂情報等を別途記述する。これは生成するソフトの Honda 仕様としての情報を含むだけでなく、ドキュメント生成結果を従来の仕様書相当とするためにも重要となる。そこで汎用の整数型ソフト自動生成ツールで用意されている、モデル内に配置した単位にソフトを分割生成する関数定義ブロックに、上記情報を記述する機能を作成した。

ソフトの自動生成は汎用の整数型ソフト自動生成ツールによる標準的書式の生成をおこなった後、本システムで付加した情報を基に、以下の処理をおこなうことでHonda仕様のソフトとする。

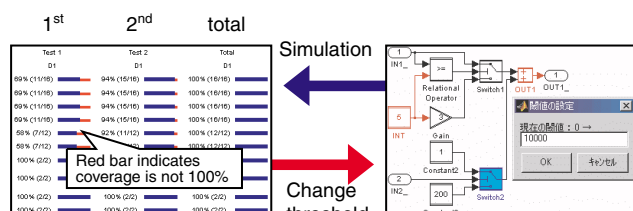


Fig. 10 Verification of coverage

Table 1 Result of coverage tool

Mode	Measurement	Test vector		
		All patterns	DOE	PWM
Number of step times	27 500	1 540	96	2 470
Coverage [%]	95	60	58	60
Number of detected overflows	0	6	5	8
Detected overflows per step time	0	0.39	5.21	0.32

(1)モデル情報(作者、作成日等)の記述

(2)変数宣言修正

(3)初期化関数記述

(4)ライブラリ登録用変数情報ファイル生成

生成したソフトを実装する場合に問題となる、ハンドコーディングしたソフトに対するサイズと実行速度について比較をおこなった。既存のソフトが存在する仕様をモデル図で作成し自動生成をおこなった結果、ベクトル処理を除いた平均ではサイズで 1.8 倍、実行速度で 1.1 倍となり、そのまま量産 ECU に適用できるレベルにあることが確認できた。

6. ドキュメント生成

従来は仕様書を基にモデル図でロジック検証をおこない、これをハンドコーディングするのが一連の流れであったが、本手法ではモデル図主導となり、仕様書が存在しない形となる。しかし前述の通りライブラリでは仕様書を含めたデータベース化をおこなっているため、登録には仕様書が必要となる。そのためモデル図によるロジック作成においても別途仕様書を作成する必要性が生じるが、これは工数的に現実的ではない。汎用の整数型ソフト自動生成ツールはモデル図を html ファイルにて出力する機能を有している。そこでこれを仕様書相当とする検討をおこなったが、汎用の整数型ソフト自動生成ツールの html ファイル出力機能は、基本的には特定の上位階層と関数定義ブロックが配置された階層のみ出力される。そこでモデル構造の把握を容易にするため、以下の開発をおこなった。

(1)配置した任意の階層も出力される出力指示ブロック

(2)ブロック図の上位階層からモデル図を順次表示

(3)作者情報等を関数定義ブロック単位で、モデル図間に記述

Fig. 11 にドキュメント出力例を示す。

モデル図をドキュメント化する際の注意点には、可読性の問題としてモデルの縮尺率、端子線の重なり等が挙げられる。これは J-MAAB(Japan MATLAB Advisory Board)においてブロック図によるモデル記述法に対するガイドラインが検討されているため、汎用の整数型ソフト自動生成

ツールを用いることに対するHonda独自の拡張をおこなったガイドラインを作成し、これに準じて作図、ドキュメント生成をおこなうこととした。

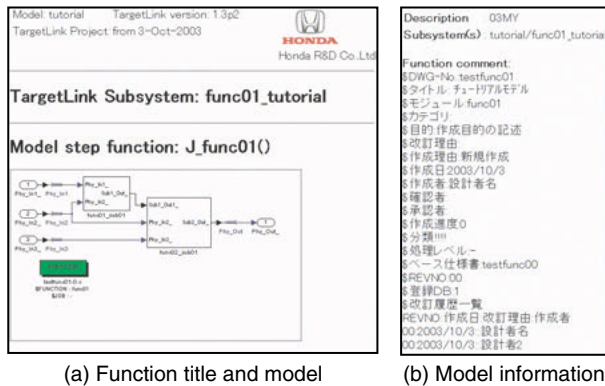


Fig. 11 Spec. sheet produced by auto document generation

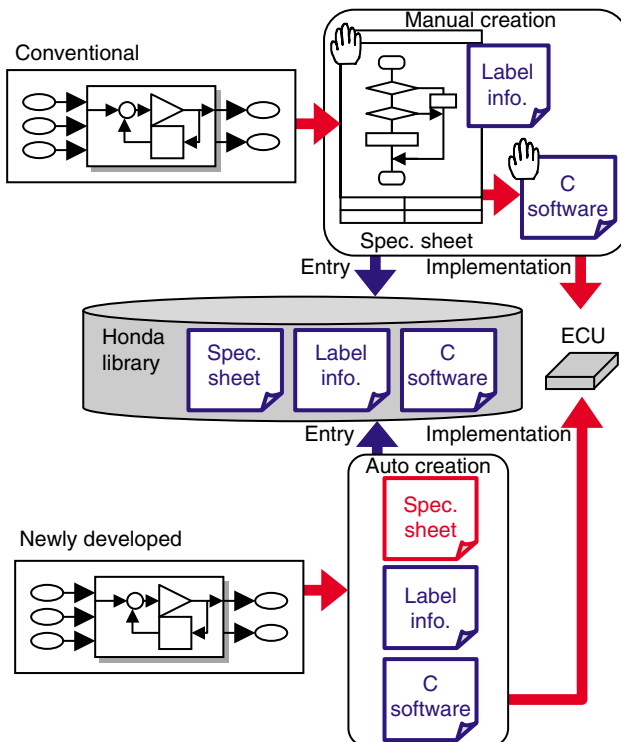


Fig. 12 Comparison of flows of model data up-load

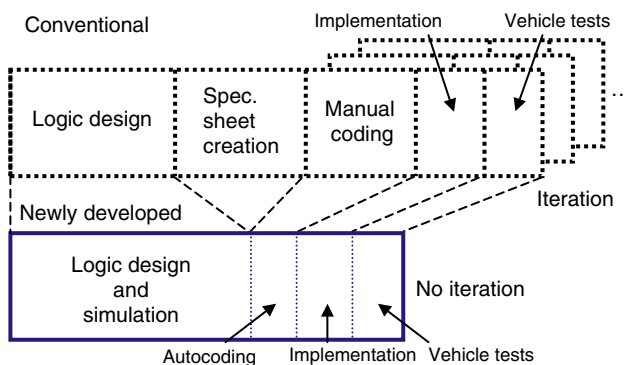


Fig. 13 Reduction of flow of design logic and load to ECU

Fig. 12にライブラリ登録フローの比較を示す。ドキュメントの自動生成化により更なる工数削減を図ることができ、ライブラリに登録する情報を全て自動で出力できた。これによりロジック構築から ECU 実装までの期間は従来に比して3分の2とすることができた。Fig. 13 に ECU 実装期間うちわけの比較を示す。

7. まとめ

本システムの開発の結果、以下の結果を得た。

- (1) モデル図より自動生成したソフトに一切手修正を加えることなく ECU に実装することができた。
- (2) 1 回の実装でシミュレーションと同等の動作を実現することができた。
- (3) 従来は実装後に発生していた戻り工数が無くなったことにより、ロジック開発から ECU 実装までの期間を3分の2に短縮することができた。

参考文献

- (1) The MathWorks, Inc. : Using Simulink Version 5, USA, The MathWorks, Inc. , 476p.(2002)
- (2) dSPACE GmbH: TargetLink Production Code Generation Guide For TargetLink 1.3, Germany, dSPACE GmbH, 748p.(2001)
- (3) 中村 義作: よくわかる実験計画法, 東京, 近代科学社, p.82-108(1997)

■ 著 者 ■



箕輪信太郎



廣田 俊明



横山 浩紀



石黒 哲矢



大塚 訓